Motivation:

In this Jupyter notebook, Kaggle data of apps of Google Play Store is used, https://www.kaggle.com/lava18/google-play-store-apps. This data comes from iTunes App Store page. While many public datasets (on Kaggle and the like) provide Apple App Store data, there are not many counterpart datasets available for Google Play Store apps anywhere on the web. Each app (row) has values for catergory, rating, size, and more.

This Jupyter notebook contains the following sections--

1, Creation of context

-creating dataframe in pyspark context

2, Preparation of data (get it into parquet)

-parquets are saved in "/home/ec2-user/SageMaker/sl4bz/tmp-pqt1"

3, Machine Learning Libraries based analysis

-Pearson's R

-Vectorization

-Training with dataset, predicting with log model and evaluating the model using R square and MSE

4, Visualization of results

-produced three plots: distribution of Rating

-Square feet Living VS Price.log

-Pairwise correlation matrix

Code Snippets:

In [20]:
```
from pyspark.sql.functions import log
from pyspark.sql.functions import col
df = df.withColumn('slpsa', df.sqft_living / df.sqft_above)
df = df.withColumn('lprice', log(df.price))
df = df.drop('price')
df.printSchema()
```

```
root
 |-- bedrooms: long (nullable = true)
 |-- bathrooms: double (nullable = true)
 |-- sqft_living: long (nullable = true)
 |-- sqft_lot: long (nullable = true)
 |-- floors: double (nullable = true)
 |-- waterfront: long (nullable = true)
 |-- view: long (nullable = true)
 |-- condition: long (nullable = true)
 |-- grade: long (nullable = true)
 |-- sqft_above: long (nullable = true)
 |-- sqft_basement: long (nullable = true)
 |-- yr_built: long (nullable = true)
 |-- yr_renovated: long (nullable = true)
 |-- lat: double (nullable = true)
 |-- long: double (nullable = true)
 |-- sqft_living15: long (nullable = true)
 |-- sqft_lot15: long (nullable = true)
 |-- slpsa: double (nullable = true)
 |-- lprice: double (nullable = true)
```

VECTORIZATION - spark special sauce

4]:
```
from pyspark.ml.linalg import Vectors, VectorUDT # nb: bad form, done for pedagogy
```

5]:
```
# make a user defined function (udf)
sqlc.registerFunction("oneElementVec", lambda d: Vectors.dense([d]), returnType=VectorUDT())

# vectorize the data frames
trainingDF = trainingDF.selectExpr("lprice", "oneElementVec(sqft_living) as sqft_living")
testDF = testDF.selectExpr("lprice", "oneElementVec(sqft_living) as sqft_living")

print(testDF.orderBy(testDF.lprice.desc()).limit(5))
```

```
DataFrame[lprice: double, sqft_living: vector]
```

6]:
```
# rename to make ML engine happy, for linear regression they have to be "label","features",dense vectors
trainingDF = trainingDF.withColumnRenamed("lprice", "label").withColumnRenamed("sqft_living", "features")
testDF = testDF.withColumnRenamed("lprice", "label").withColumnRenamed("sqft_living", "features")
```

]:
```
from pyspark.ml.regression import LinearRegression, LinearRegressionModel

lr = LinearRegression()
lrModel = lr.fit(trainingDF)
```

]:
```
type(lrModel)
```

]:
```
pyspark.ml.regression.LinearRegressionModel
```

]:
```
predictionsAndLabelsDF = lrModel.transform(testDF)

print(predictionsAndLabelsDF.orderBy(predictionsAndLabelsDF.label.desc()).take(5))
```

```
[Row(label=15.483217378522351, features=DenseVector([7390.0]), prediction=15.160691991443286), Row(label=15.35666962964508, features=DenseVector([9640.0]), prediction=16.0568277966743), Row(label=15.252973205658565, features=DenseVector([7440.0]), prediction=15.18060612044842), Row(label=15.126542434583618, features=DenseVector([5550.0]), prediction=14.427852044054372), Row(label=15.110237725558674, features=DenseVector([5020.0]), prediction=14.216762276599956)]
```

Model Evaluation

]:
```
from pyspark.ml.evaluation import RegressionEvaluator
eval = RegressionEvaluator()
print(eval.explainParams())
```

```
labelCol: label column name. (default: label)
```

Visualizations: